FTPM Organization

Analysis for more modularity

Sat, Apr 14, 2001

The `NServer` routine in the `FTPM` local application initializes a non-server request for a Fast Time Plot protocol request. This routine requires 9 pages of source code in Pascal. It includes support for a number of types of digitizers. With such extensive examples, it should be possible to design an organization that could make adding yet another digitizer an easier job. The source code breaks out into separate code for each digitizer type, so one approach could divide these pieces of the code into separate modules called by `NServer`.

The separate portions of `NServer` break down into the several protocols whose support is covered by `FTPM`. These are called `timing1`, `continuous6`, and `snapshot7`. `Timing1` supports the initial "what can you do for this device?" request. The `continuous6` protocol supports collection of a continuous stream of "fast time plot" data, where contributions of the most recent measurements are returned to the requester at periods from 1–7 cycles at 15Hz. The `snapshot7` protocol covers waveform captures from digitizer hardware. The latter logic further breaks down into support for a number of different digitizers.

The `timing1` logic determines which class codes should be returned to implicitely describe the support available for a possible subsequent `continuous6` request or a `snapshot7` request. For the `continuous6` case, there is always the default level of support of 15 Hz, since the entire data pool is updated at that rate. Assuming that the listype indicated in the SSDN is `0x00`, meaning analog reading, `FTDChan` is invoked to determine whether up to 1KHz continuous data can be collected from the given channel. Those are the only choices for continuous plots. The snapshot case is a bit trickier. If a valid Swift/Quicker (IP board) `CINFO` table entry exists for the given channel, one of two snapshot class types is returned. If a valid Quick `CINFO` table entry exists, then return the code for the Quick (commercial VME board) digitizer. The last case is to check for a valid KHz (IRM 1 KHz) `CINFO` entry. If there is, such data can be captured into a snapshot buffer; otherwise, ordinary 15 Hz data pool access can be supported, even if it might take some time to accumulate a complete waveform at such a slow rate.

The `continuous6` logic is short. It only prepares for the first-time reply to the `continuous6` request, which means it does not include the data. For the case of sampling from the 1KHz circular buffer memory, the `internalPtr` array is set, preparing for the time when `RFTData` is called to collect the data.

The `snapshot7` logic is not short. It includes most of the code in `NServer`. It sets the `internalPtr` array element to the appropriate waveform base address, depending on which fast digitizer type applies. Then special logic applies for each case. For the Swift case, there are two variations: auto-triggered and not. In each variation, the fields of the periodic status reply message are determined. If the hardware is auto-triggered, the registers are read to set the reply fields; otherwise, the request fields are interpreted to match up with the ability of the digitizer. For the Quick case, the registers determine the reply fields. For the kHz case, there are again two variations: the 15Hz cycle case and the general case of any rate up to 1000 Hz.